National Textile University

Department of Computer Science

Subject:

Operating System

Submitted to:

Sir Nasir

Submitted by:

Eman Babar

Reg. number:

23-NTU-CS-FL-1148

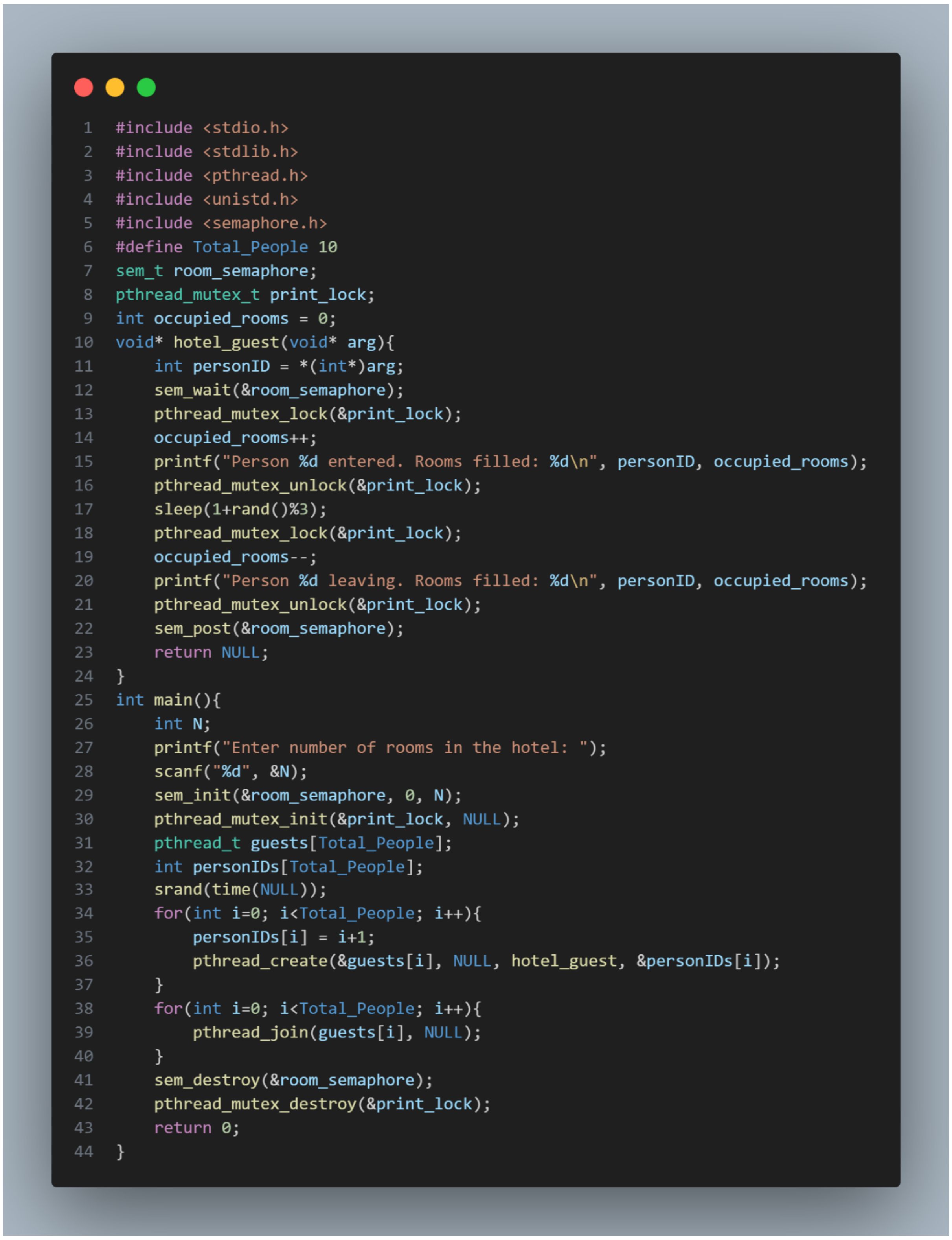
Semester:

5th- A

LAB-10-HomeTask

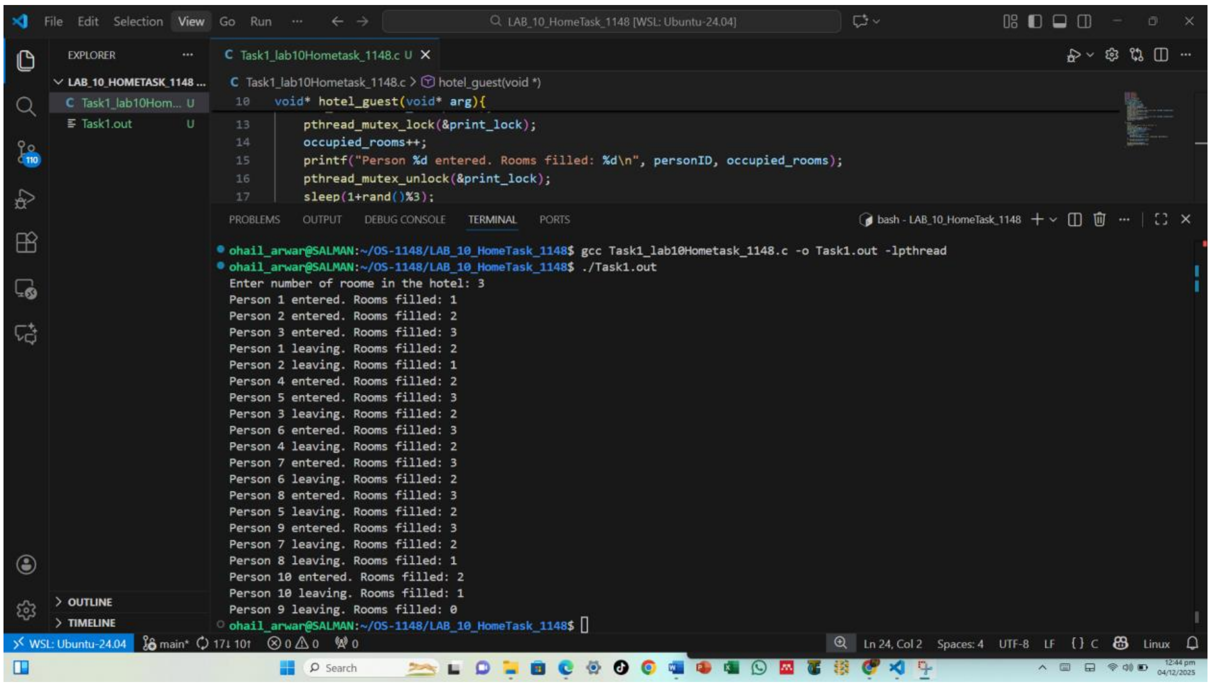
Task1: Hotel Rom Occupancy Problem

Code:



```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>
#define Total_People 10
sem_t room_semaphore;
pthread_mutex_t print_lock;
int occupied_rooms = 0;
void* hotel_guest(void* arg){
    int personID = *(int*)arg;
    sem_wait(&room_semaphore);
    pthread_mutex_lock(&print_lock);
    occupied_rooms++;
    printf("Person %d entered. Rooms filled: %d\n", personID, occupied_rooms);
    pthread_mutex_unlock(&print_lock);
    sleep(1+rand()%3);
    pthread_mutex_lock(&print_lock);
    occupied_rooms--;
    printf("Person %d leaving. Rooms filled: %d\n", personID, occupied_rooms);
    pthread_mutex_unlock(&print_lock);
    sem_post(&room_semaphore);
    return NULL;
}
int main(){
    int N;
    printf("Enter number of rooms in the hotel: ");
    scanf("%d", &N);
    sem_init(&room_semaphore, 0, N);
    pthread_mutex_init(&print_lock, NULL);
    pthread_t guests[Total_People];
    int personIDs[Total_People];
    srand(time(NULL));
    for(int i=0; i<Total_People; i++){
        personIDs[i] = i+1;
        pthread_create(&guests[i], NULL, hotel_guest, &personIDs[i]);
    }
    for(int i=0; i<Total_People; i++){
        pthread_join(guests[i], NULL);
    }
    sem_destroy(&room_semaphore);
    pthread_mutex_destroy(&print_lock);
    return 0;
}
```

Output:

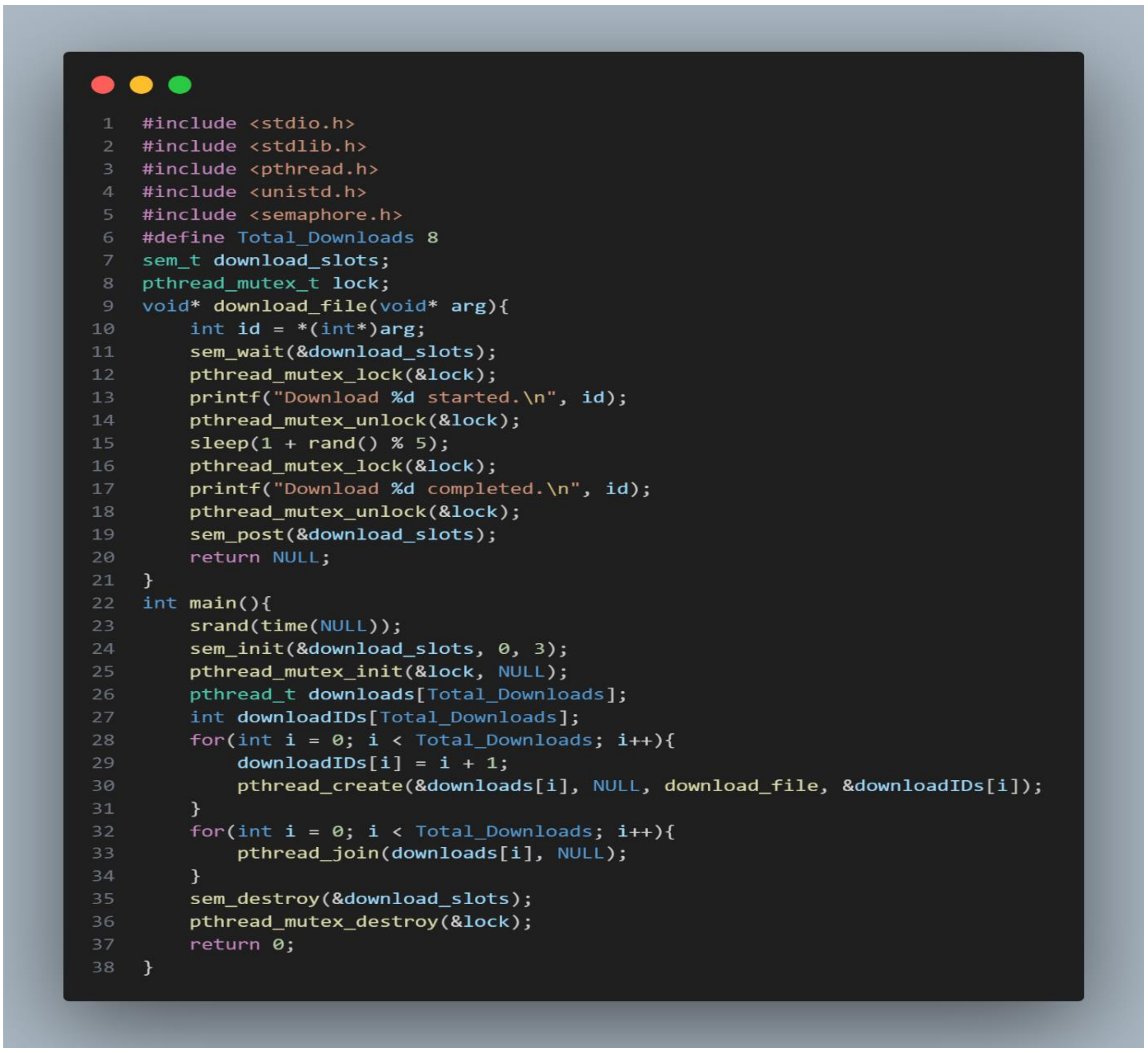


```c
void* hotel_guest(void* arg){
    pthread_mutex_lock(&print_lock);
    occupied_rooms++;
    printf("Person %d entered. Rooms filled: %d\n", personID, occupied_rooms);
    pthread_mutex_unlock(&print_lock);
    sleep(1+rand()%3);
```

```
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ gcc Task1_lab10Hometask_1148.c -o Task1.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ ./Task1.out
Enter number of roome in the hotel: 3
Person 1 entered. Rooms filled: 1
Person 2 entered. Rooms filled: 2
Person 3 entered. Rooms filled: 3
Person 1 leaving. Rooms filled: 2
Person 2 leaving. Rooms filled: 1
Person 4 entered. Rooms filled: 2
Person 5 entered. Rooms filled: 3
Person 3 leaving. Rooms filled: 2
Person 6 entered. Rooms filled: 3
Person 4 leaving. Rooms filled: 2
Person 7 entered. Rooms filled: 3
Person 6 leaving. Rooms filled: 2
Person 8 entered. Rooms filled: 3
Person 5 leaving. Rooms filled: 2
Person 9 entered. Rooms filled: 3
Person 7 leaving. Rooms filled: 2
Person 8 leaving. Rooms filled: 1
Person 10 entered. Rooms filled: 2
Person 10 leaving. Rooms filled: 1
Person 9 leaving. Rooms filled: 0
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$
```

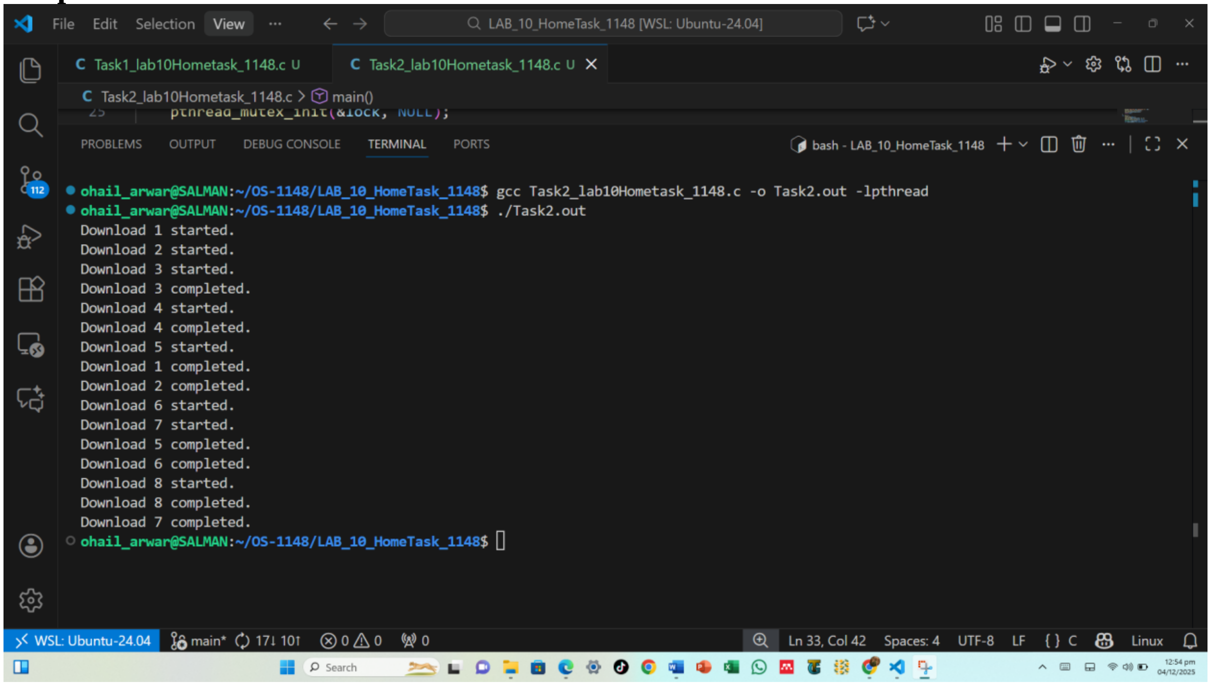
Task2: Download Manager Simulation

Code:



```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>
#define Total_Downloads 8
sem_t download_slots;
pthread_mutex_t lock;
void* download_file(void* arg){
    int id = *(int*)arg;
    sem_wait(&download_slots);
    pthread_mutex_lock(&lock);
    printf("Download %d started.\n", id);
    pthread_mutex_unlock(&lock);
    sleep(1 + rand() % 5);
    pthread_mutex_lock(&lock);
    printf("Download %d completed.\n", id);
    pthread_mutex_unlock(&lock);
    sem_post(&download_slots);
    return NULL;
}
int main(){
    srand(time(NULL));
    sem_init(&download_slots, 0, 3);
    pthread_mutex_init(&lock, NULL);
    pthread_t downloads[Total_Downloads];
    int downloadIDs[Total_Downloads];
    for(int i = 0; i < Total_Downloads; i++){
        downloadIDs[i] = i + 1;
        pthread_create(&downloads[i], NULL, download_file, &downloadIDs[i]);
    }
    for(int i = 0; i < Total_Downloads; i++){
        pthread_join(downloads[i], NULL);
    }
    sem_destroy(&download_slots);
    pthread_mutex_destroy(&lock);
    return 0;
}
```

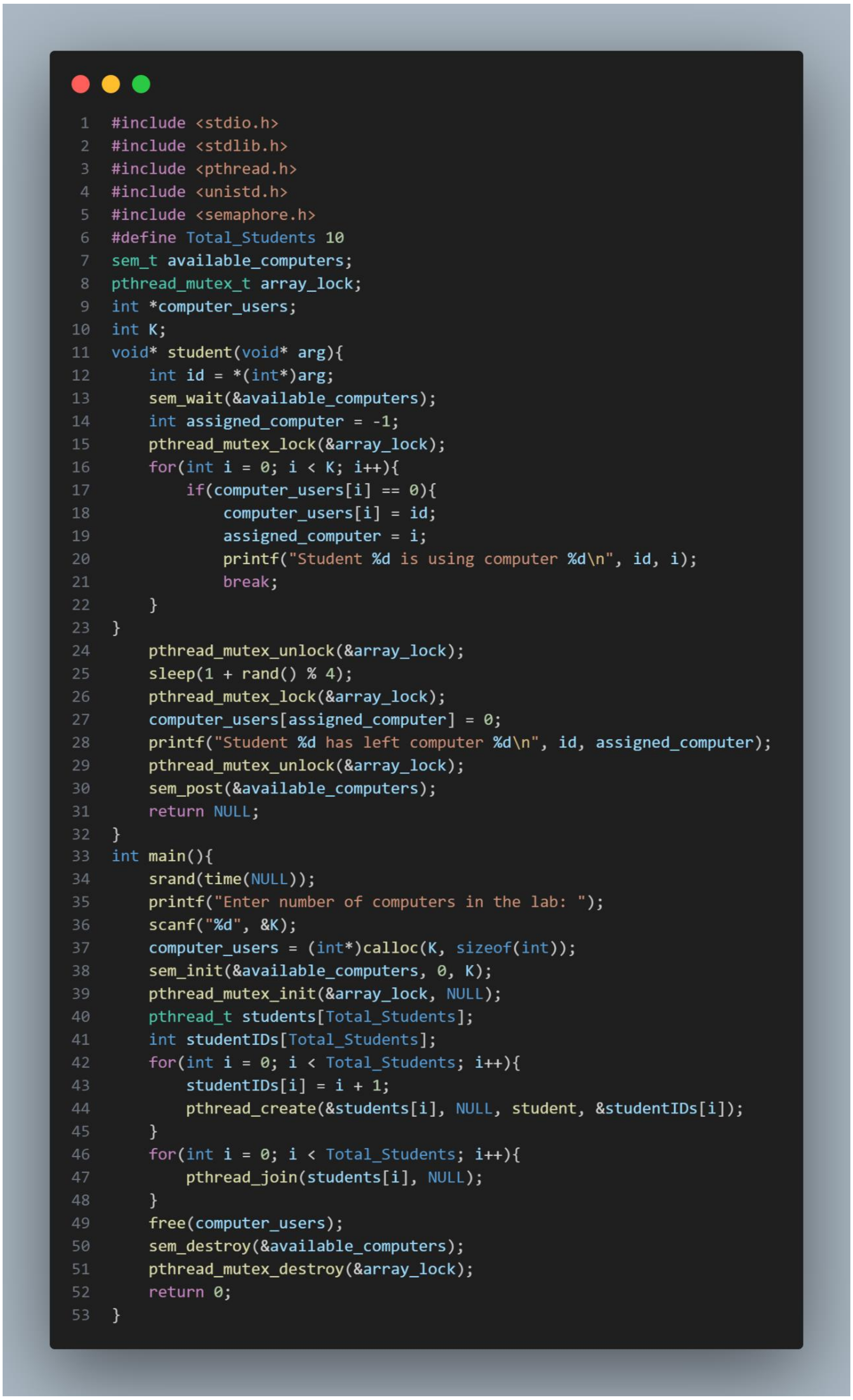
Output:



```
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ gcc Task2_lab10Hometask_1148.c -o Task2.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ ./Task2.out
Download 1 started.
Download 2 started.
Download 3 started.
Download 3 completed.
Download 4 started.
Download 4 completed.
Download 5 started.
Download 1 completed.
Download 2 completed.
Download 6 started.
Download 7 started.
Download 5 completed.
Download 6 completed.
Download 8 started.
Download 8 completed.
Download 7 completed.
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$
```

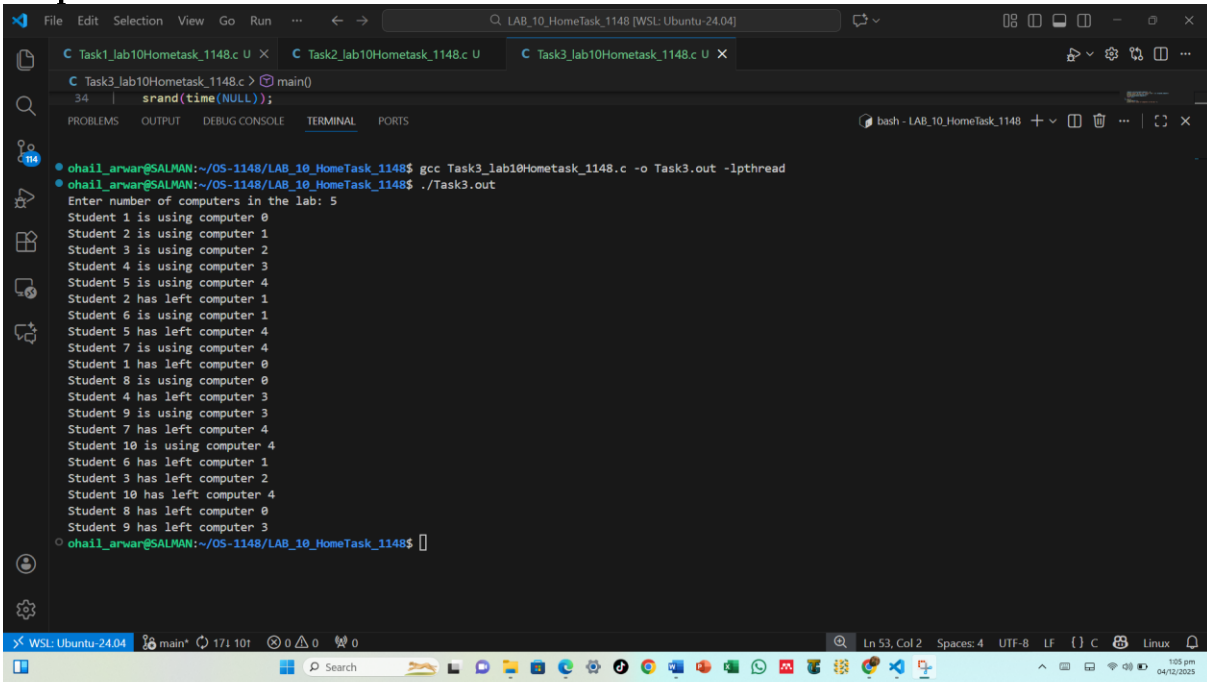
<u>Task3: Library Computer Lab Access</u>

Code:



```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>
#define Total_Students 10
sem_t available_computers;
pthread_mutex_t array_lock;
int *computer_users;
int K;
void* student(void* arg){
    int id = *(int*)arg;
    sem_wait(&available_computers);
    int assigned_computer = -1;
    pthread_mutex_lock(&array_lock);
    for(int i = 0; i < K; i++){
        if(computer_users[i] == 0){
            computer_users[i] = id;
            assigned_computer = i;
            printf("Student %d is using computer %d\n", id, i);
            break;
    }
}
    pthread_mutex_unlock(&array_lock);
    sleep(1 + rand() % 4);
    pthread_mutex_lock(&array_lock);
    computer_users[assigned_computer] = 0;
    printf("Student %d has left computer %d\n", id, assigned_computer);
    pthread_mutex_unlock(&array_lock);
    sem_post(&available_computers);
    return NULL;
}
int main(){
    srand(time(NULL));
    printf("Enter number of computers in the lab: ");
    scanf("%d", &K);
    computer_users = (int*)calloc(K, sizeof(int));
    sem_init(&available_computers, 0, K);
    pthread_mutex_init(&array_lock, NULL);
    pthread_t students[Total_Students];
    int studentIDs[Total_Students];
    for(int i = 0; i < Total_Students; i++){
        studentIDs[i] = i + 1;
        pthread_create(&students[i], NULL, student, &studentIDs[i]);
    }
    for(int i = 0; i < Total_Students; i++){
        pthread_join(students[i], NULL);
    }
    free(computer_users);
    sem_destroy(&available_computers);
    pthread_mutex_destroy(&array_lock);
    return 0;
}
```

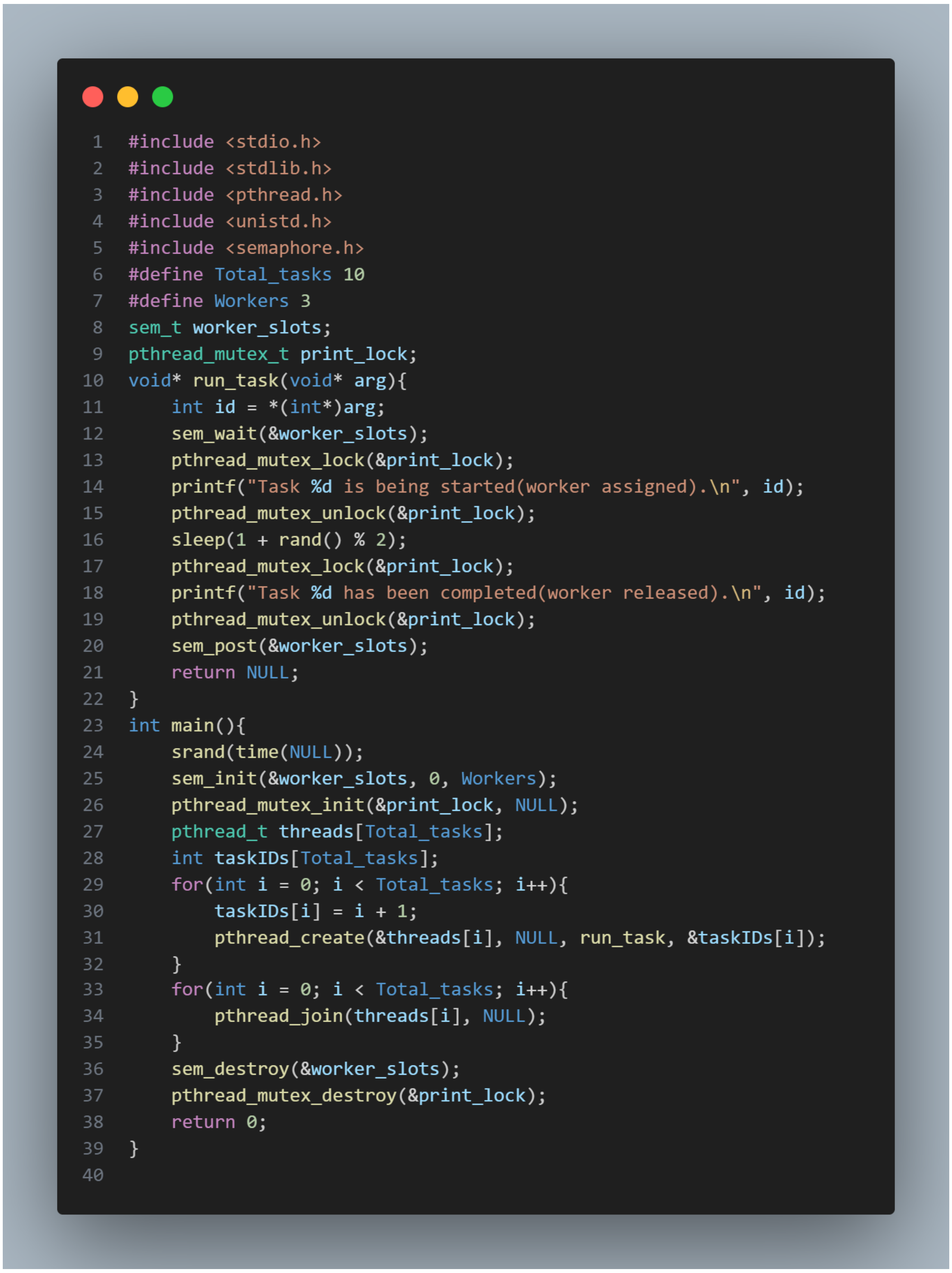
Output:



```
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ gcc Task3_lab10Hometask_1148.c -o Task3.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ ./Task3.out
Enter number of computers in the lab: 5
Student 1 is using computer 0
Student 2 is using computer 1
Student 3 is using computer 2
Student 4 is using computer 3
Student 5 is using computer 4
Student 2 has left computer 1
Student 6 is using computer 1
Student 5 has left computer 4
Student 7 is using computer 4
Student 1 has left computer 0
Student 8 is using computer 0
Student 4 has left computer 3
Student 9 is using computer 3
Student 7 has left computer 4
Student 10 is using computer 4
Student 6 has left computer 1
Student 3 has left computer 2
Student 10 has left computer 4
Student 8 has left computer 0
Student 9 has left computer 3
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ 
```

<u>Task4: Thread Pool/ Worker Pool Simulation</u>

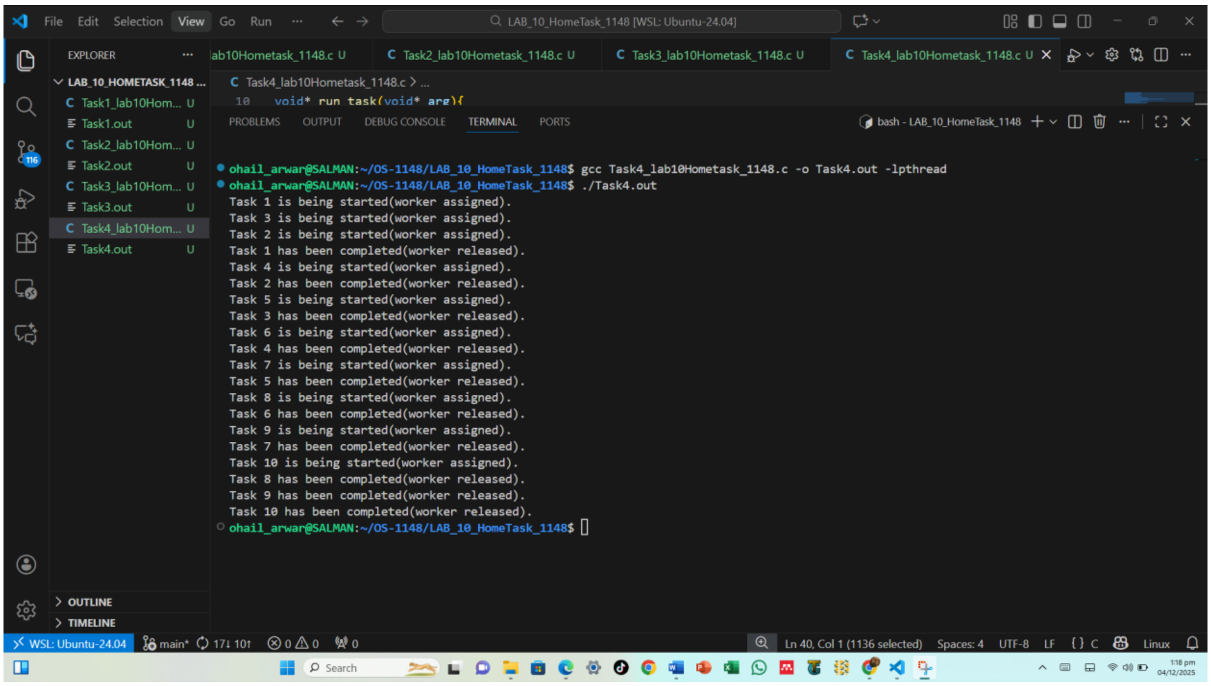
Code:



```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>
#define Total_tasks 10
#define Workers 3
sem_t worker_slots;
pthread_mutex_t print_lock;
void* run_task(void* arg){
    int id = *(int*)arg;
    sem_wait(&worker_slots);
    pthread_mutex_lock(&print_lock);
    printf("Task %d is being started(worker assigned).\n", id);
    pthread_mutex_unlock(&print_lock);
    sleep(1 + rand() % 2);
    pthread_mutex_lock(&print_lock);
    printf("Task %d has been completed(worker released).\n", id);
    pthread_mutex_unlock(&print_lock);
    sem_post(&worker_slots);
    return NULL;
}
int main(){
    srand(time(NULL));
    sem_init(&worker_slots, 0, Workers);
    pthread_mutex_init(&print_lock, NULL);
    pthread_t threads[Total_tasks];
    int taskIDs[Total_tasks];
    for(int i = 0; i < Total_tasks; i++){
        taskIDs[i] = i + 1;
        pthread_create(&threads[i], NULL, run_task, &taskIDs[i]);
    }
    for(int i = 0; i < Total_tasks; i++){
        pthread_join(threads[i], NULL);
    }
    sem_destroy(&worker_slots);
    pthread_mutex_destroy(&print_lock);
    return 0;
}

```

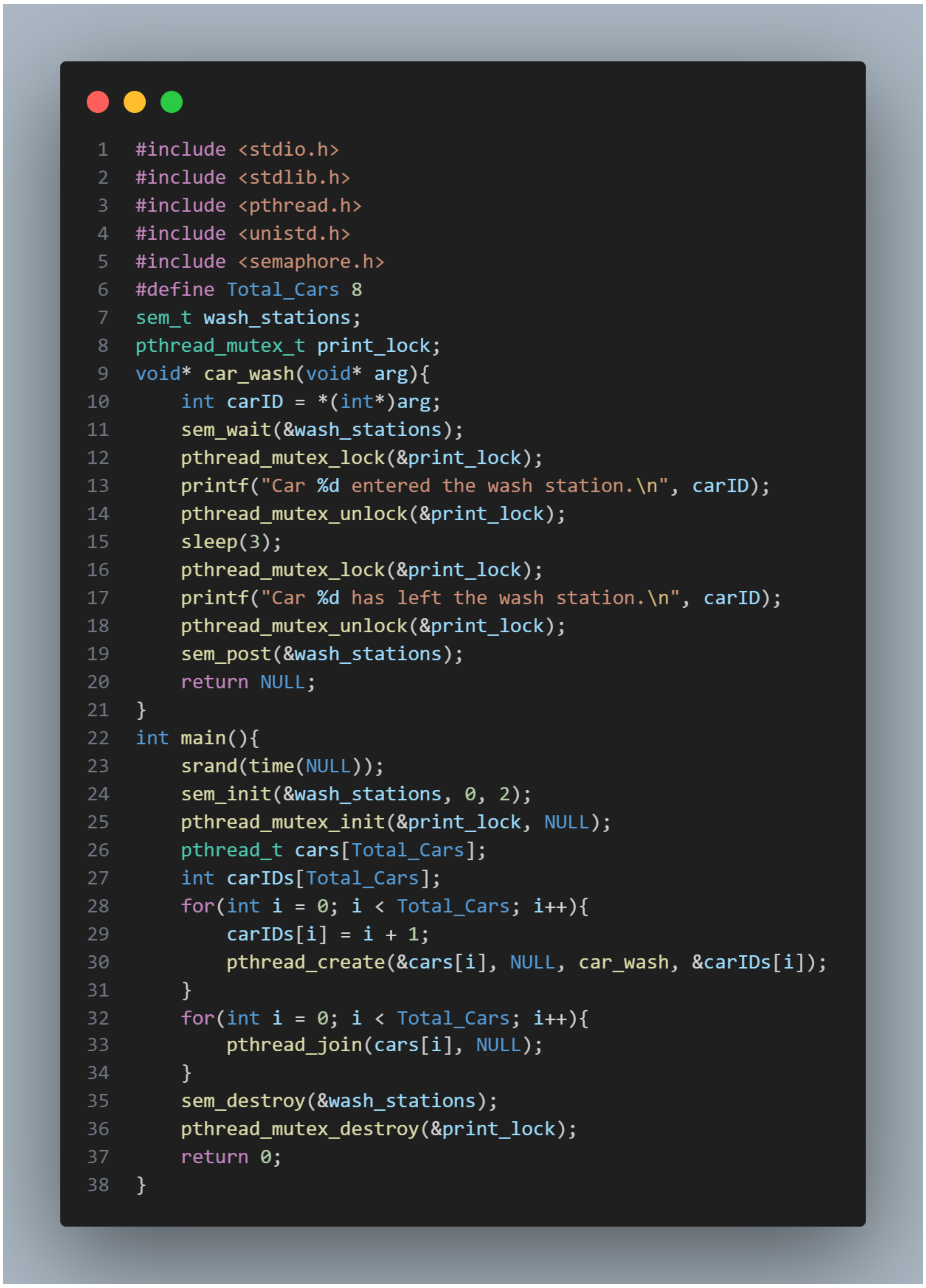
Output:



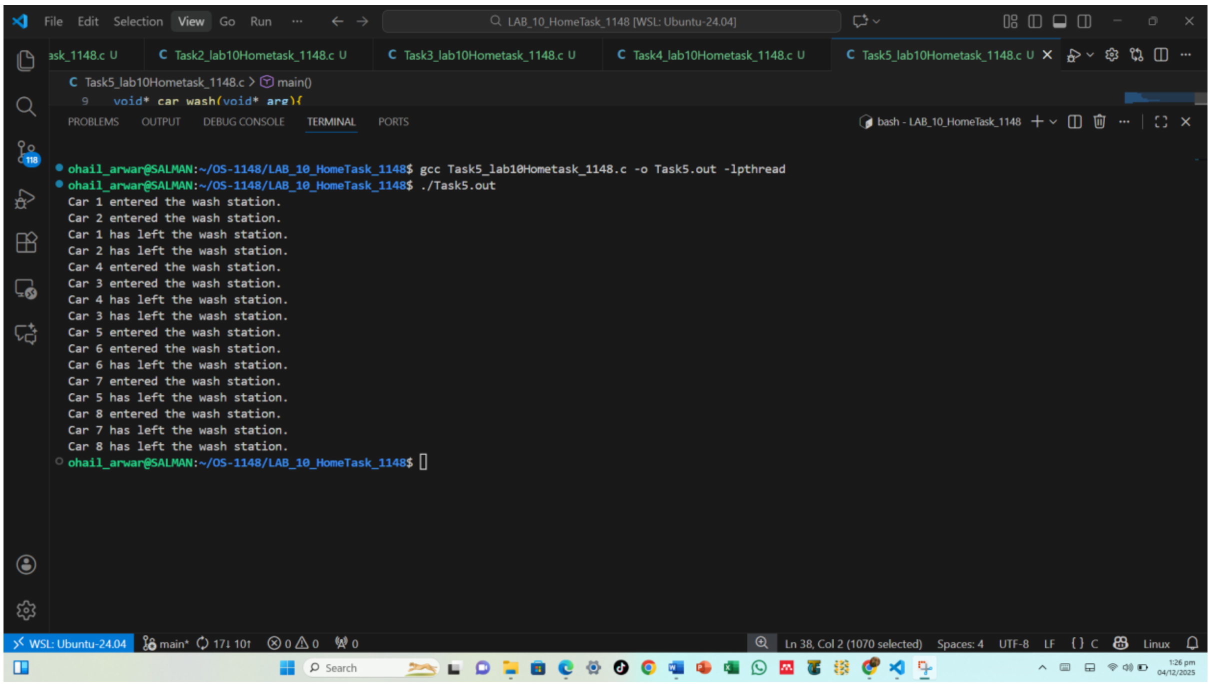
```
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ gcc Task4_lab10Hometask_1148.c -o Task4.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ ./Task4.out
Task 1 is being started(worker assigned).
Task 3 is being started(worker assigned).
Task 2 is being started(worker assigned).
Task 1 has been completed(worker released).
Task 4 is being started(worker assigned).
Task 2 has been completed(worker released).
Task 5 is being started(worker assigned).
Task 3 has been completed(worker released).
Task 6 is being started(worker assigned).
Task 4 has been completed(worker released).
Task 7 is being started(worker assigned).
Task 5 has been completed(worker released).
Task 8 is being started(worker assigned).
Task 6 has been completed(worker released).
Task 9 is being started(worker assigned).
Task 7 has been completed(worker released).
Task 10 is being started(worker assigned).
Task 8 has been completed(worker released).
Task 9 has been completed(worker released).
Task 10 has been completed(worker released).
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ 
```

Task5: Car Wash Station

Code:


```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>
#define Total_Cars 8
sem_t wash_stations;
pthread_mutex_t print_lock;
void* car_wash(void* arg){
    int carID = *(int*)arg;
    sem_wait(&wash_stations);
    pthread_mutex_lock(&print_lock);
    printf("Car %d entered the wash station.\n", carID);
    pthread_mutex_unlock(&print_lock);
    sleep(3);
    pthread_mutex_lock(&print_lock);
    printf("Car %d has left the wash station.\n", carID);
    pthread_mutex_unlock(&print_lock);
    sem_post(&wash_stations);
    return NULL;
}
int main(){
    srand(time(NULL));
    sem_init(&wash_stations, 0, 2);
    pthread_mutex_init(&print_lock, NULL);
    pthread_t cars[Total_Cars];
    int carIDs[Total_Cars];
    for(int i = 0; i < Total_Cars; i++){
        carIDs[i] = i + 1;
        pthread_create(&cars[i], NULL, car_wash, &carIDs[i]);
    }
    for(int i = 0; i < Total_Cars; i++){
        pthread_join(cars[i], NULL);
    }
    sem_destroy(&wash_stations);
    pthread_mutex_destroy(&print_lock);
    return 0;
}
```

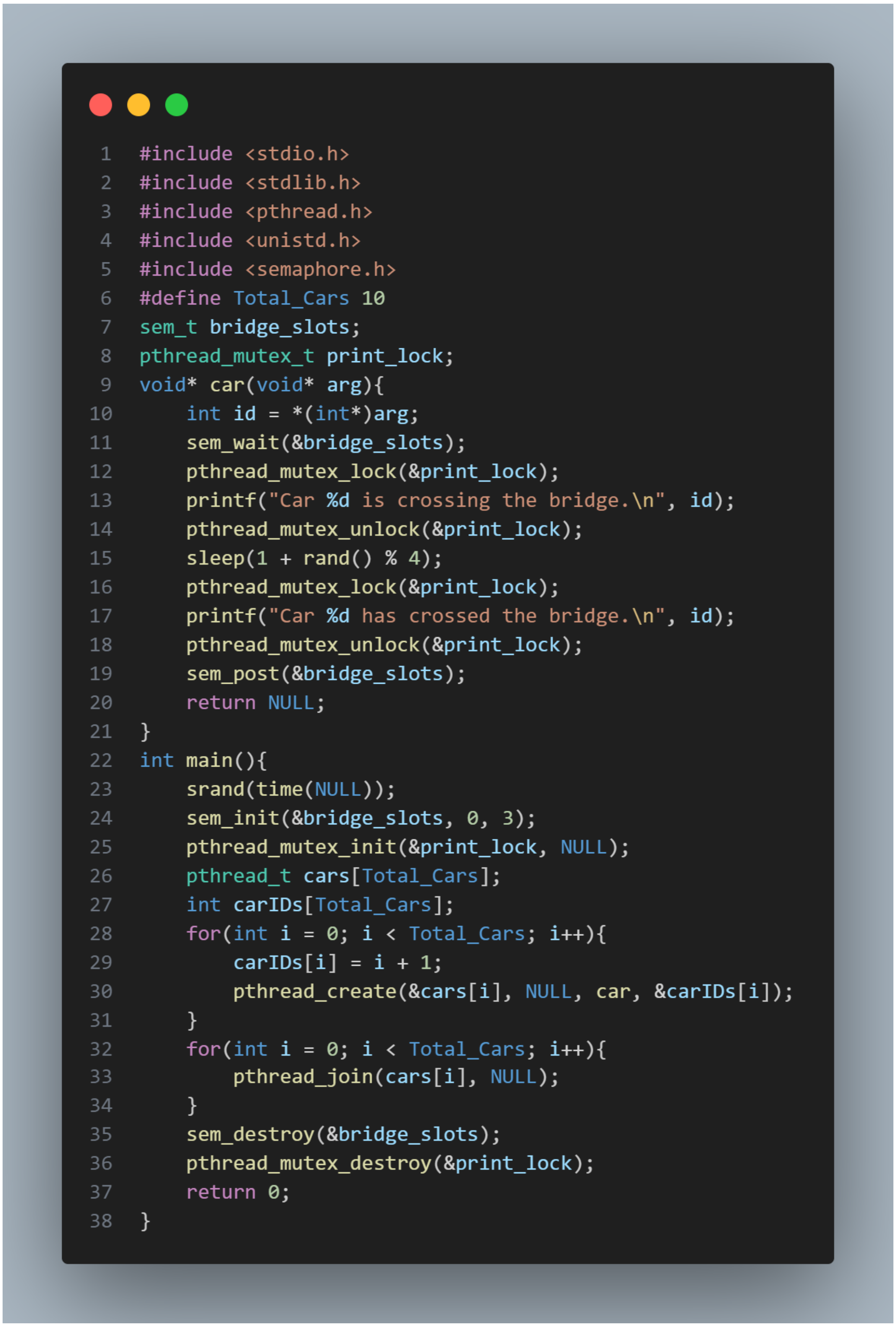
Output:



```
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ gcc Task5_lab10Hometask_1148.c -o Task5.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ ./Task5.out
Car 1 entered the wash station.
Car 2 entered the wash station.
Car 1 has left the wash station.
Car 2 has left the wash station.
Car 4 entered the wash station.
Car 3 entered the wash station.
Car 4 has left the wash station.
Car 3 has left the wash station.
Car 5 entered the wash station.
Car 6 entered the wash station.
Car 6 has left the wash station.
Car 7 entered the wash station.
Car 5 has left the wash station.
Car 8 entered the wash station.
Car 7 has left the wash station.
Car 8 has left the wash station.
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ 
```

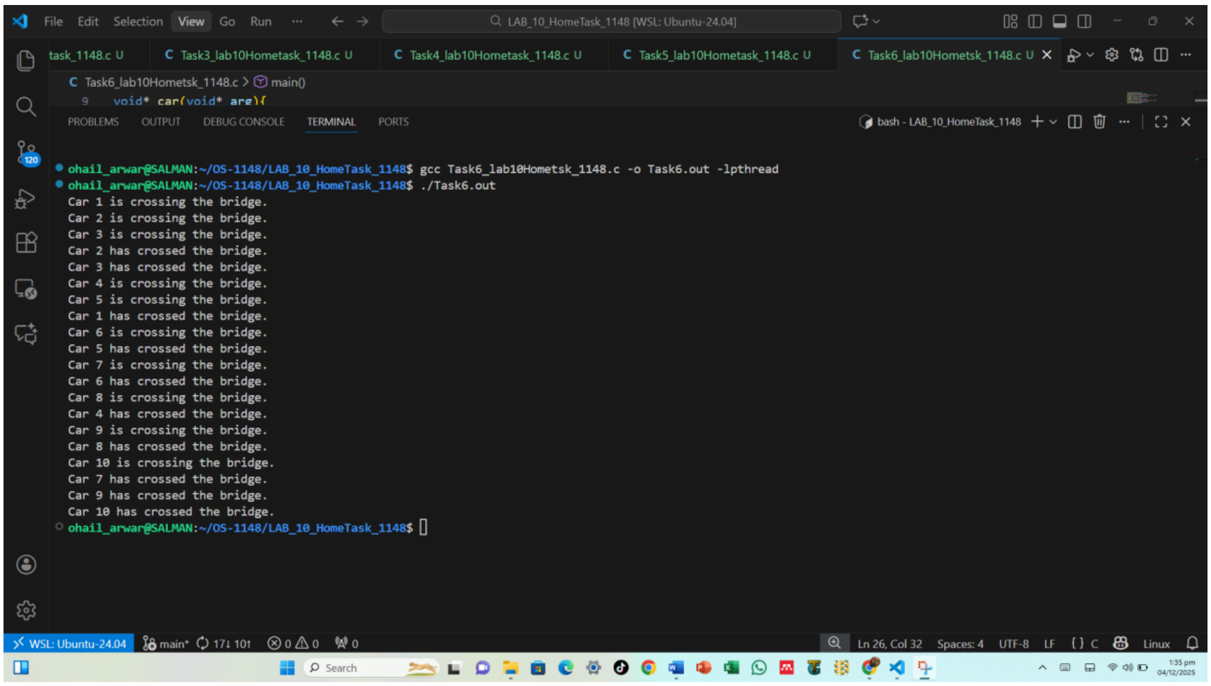
<u>Task6: Traffic Bridge Control(Single-Lane Bridge)</u>

Code:



```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>
#define Total_Cars 10
sem_t bridge_slots;
pthread_mutex_t print_lock;
void* car(void* arg){
    int id = *(int*)arg;
    sem_wait(&bridge_slots);
    pthread_mutex_lock(&print_lock);
    printf("Car %d is crossing the bridge.\n", id);
    pthread_mutex_unlock(&print_lock);
    sleep(1 + rand() % 4);
    pthread_mutex_lock(&print_lock);
    printf("Car %d has crossed the bridge.\n", id);
    pthread_mutex_unlock(&print_lock);
    sem_post(&bridge_slots);
    return NULL;
}
int main(){
    srand(time(NULL));
    sem_init(&bridge_slots, 0, 3);
    pthread_mutex_init(&print_lock, NULL);
    pthread_t cars[Total_Cars];
    int carIDs[Total_Cars];
    for(int i = 0; i < Total_Cars; i++){
        carIDs[i] = i + 1;
        pthread_create(&cars[i], NULL, car, &carIDs[i]);
    }
    for(int i = 0; i < Total_Cars; i++){
        pthread_join(cars[i], NULL);
    }
    sem_destroy(&bridge_slots);
    pthread_mutex_destroy(&print_lock);
    return 0;
}
```

Output:



```
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ gcc Task6_lab10Hometsk_1148.c -o Task6.out -lpthread
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$ ./Task6.out
Car 1 is crossing the bridge.
Car 2 is crossing the bridge.
Car 3 is crossing the bridge.
Car 2 has crossed the bridge.
Car 3 has crossed the bridge.
Car 4 is crossing the bridge.
Car 5 is crossing the bridge.
Car 1 has crossed the bridge.
Car 6 is crossing the bridge.
Car 5 has crossed the bridge.
Car 7 is crossing the bridge.
Car 6 has crossed the bridge.
Car 8 is crossing the bridge.
Car 4 has crossed the bridge.
Car 9 is crossing the bridge.
Car 8 has crossed the bridge.
Car 10 is crossing the bridge.
Car 7 has crossed the bridge.
Car 9 has crossed the bridge.
Car 10 has crossed the bridge.
ohail_arwar@SALMAN:~/OS-1148/LAB_10_HomeTask_1148$
```